PEMANFAATAN ALGORITMA RABIN-KARP UNTUK MENGETAHUI TINGKAT KEMIRIPAN DARI SOURCE CODE PADA PEMROGRAMAN LISP

Ahmad Aulia Wiguna, Ifan Rizqa

Abstract - Programming algorithm is a subject which has to be taught in Informatics Engineering Departement Faculty of Computer Science Udinus Semarang. In daily course, students gather daily task to an assistant lecturer to be assessed. In the assessment , assistant lecturer and lecturer will not see the level of similarity between each task to each other because of there is no tool which can detect percentage of similarities between student's task. Therefore, we have to build an application that have ability to detect similarity of each student assignment. Rabin-Karp algorithm is an algorithm which can detect the similarity of each lisp source code files. How Rabin-Karp algorithm works is read each substring in a source code and calculating its hash. A source code hash's compared to each other source code's hash so that the percentage of similarity can be revealed. The result may be used as consideration in giving score to the students.

Keywords – Udinus Semarang, Programming algorithm, plagiarism Rabin-Karp algorithm, application

I. PENDAHULUAN

Dalam bidang pendidikan Teknik Informatika, pemrograman merupakan salah satu mata kuliah wajib yang diajarkan. Belajar pemrograman berbeda dengan belajar bahasa pemrograman. Belajar pemrograman adalah belajar tentang strategi, metodologi, sistematika pemecahan masalah kemudian menuliskannya dalam suatu notasi. Sedangkan belajar bahasa pemrograman adalah belajar memakai suatu bahasa, aturan sintaks, instruksi, pengoperasian *compiler* atau *interpreter* pada mesin tertentu [1]. Pembelajaran pemrograman yang terdiri dari kegiatan menyusun algoritma dan belajar memprogram selalu ada pada kurikulum program studi yang berkonsentrasi pada teknologi dan sistem informasi. Tujuan diberikannya mata kuliah Pemrograman adalah agar mahasiswa dapat membuat suatu program yang dapat melakukan perhitungan atau "pekerjaan" sesuai keinginan pemrogram. Dalam silabus dan satuan acara pengajaran Sekolah Teknik Elektro dan Informatika ITB kurikulum 2013-2018 dijelaskan bahwa mata kuliah dasar pemrograman berfungsi untuk mengenalkan konsep fundamental pemrograman: abstraksi, dekomposisi problem, modularisasi, rekurens; skill/praktek pemrograman skala kecil (aspek koding) sedangkan mata kuliah Algoritma dan Struktur Data bermanfaat untuk menanamkan pemahaman konsep algoritma dan struktur data yang umum dipakai di bidang informatika atau ilmu komputer. Saat ini sudah ada metode yang dapat digunakan untuk melacak kesamaan *source code* atau notasi algoritma. Salah satunya [3] adalah penelitian Sevtian Agus Prasetyo pada tahun 2013 berjudul Deteksi Plagiarisme Source Code Berbasis SIM Dengan Metode Perbaikan Coding Models Algorithm, penelitian tersebut menjelaskan tentang teknik deteksi kode sumber program dalam perkuliahan dasar pemrograman. Penelitian tersebut berfokus pada pelacakan persentase kesamaan antara 2 (dua) atau lebih kode sumber program bahasa C pada kuliah dasar pemrograman dan struktur data dengan menggunakan perangkat lunak SIM yang dikombinasikan dengan metode perbaikan Coding Models Algorithm.

Fokus dari penelitian ini adalah pemrograman fungsional. Pemrograman fungsional didasari oleh oleh konsep pemetaan dan fungsi matematika. Dalam paradigma ini, tidak lagi dipermasalahkan tentang alokasi memori, struktur data, *variable*. Kemudian, obyek penelitian adalah kode sumber LISP pada perkuliahan algoritma pemrograman (fungsional) di Laboratorium Dasar Fasilkom Udinus. Dalam perkuliahan, mengumpulkan tugas adalah suatu rutinitas. Dengan banyaknya mahasiswa yang mengikuti kuliah algoritma pemrograman tentunya akan banyak tugas yang dikumpulkan dimana masing-masing tugas tersebut memiliki tujuan yang sama berdasarkan penugasan. Hal ini dapat meningkatkan resiko terjadinya plagiasi tugas.

Berkas *source* code LISP yang dijadikan obyek penelitian diperlakukan sebagai berkas teks biasa. Ada beberapa metode dan algoritma yang dapat dipakai untuk mendeteksi plagiasi dokumen teks. Dalam penelitian berjudul *"Deteksi Plagiat Dokumen Menggunakan Algoritma Rabin-Karp"* oleh Hari Bagus Firdaus pada tahun 2008, dijelaskan tentang teknik mendeteksi plagiat dokumen menggunakan algoritma Rabin-Karp dimana algoritma tersebut bekerja dengan cara membandingkan nilai *hash string* masukan dan *substring* pada teks yang diuji [4]. Algoritma lain yang dapat dipakai untuk mendeteksi plagiasi adalah algoritma Winnowing dimana algoritma ini mengubah rangkaian N-Gram dokumen yang diuji menjadi kumpulan *hash*. N-Gram adalah urutan potongan kata atau karakter sejumlah n buah [5].

II. METODE YANG DIUSULKAN

Penelitian yang sebelumnya pernah dilakukan dan relevan dengan penelitian ini salah satunya adalah penelitian oleh Diana Purwitasari, Putu Yuwono Kusmawan, Umi Laili Yuhana yang berjudul "*Deteksi Keberadaan Kalimat Sama sebagai Indikasi Penjiplakan dengan Algoritma Hashing Berbasis N-Gram*". Penelitian tersebut membahas tentang cara mendeteksi kalimat yang sama sebagai hasil *copy-paste*. Menggunakan algoritma

Winnowing. Dalam penelitian tersebut ada beberapa hal dasar [5] yang harus dipenuhi oluh algoritma tersebut yaitu 1) Pencarian kalimat tidak boleh terpengaruh oleh spasi atau ruang kosong, jenis huruf, tanda baca 2) Mengabaikan kata yang pandek dan jarang digunakan 3) pencarian kesamaan kata tidak boleh terpengaruh oleh posisi karakter atau *string* dengan kata lain bahwa suatu *string* harus dapat dikenali meskipun berada pada posisi yang berbeda.

Penelitian lain yang relevan adalah yang dilakukan oleh Sevtian Agus Prasetyo berjudul "*Deteksi Plagiarisme Source Code berbasis SIM dengan Metode Perbaikan Coding Models Algorithm*". Penelitian ini berfokus pada deteksi plagiarism kode sumber bahasa C dengan menggunakan perangkat bantu SIM yang disempurnakan dengan perbaikan *coding models algorithm*. Keluaran yang dihasilkan adalah presentase kemiripan antara file kode sumber bahasa C yang diuji dengan file kode sumber bahasa C yang dijadikan contoh.

Metode yang digunakan dalam peneliian ini adalah membuat sistem deteksi plagiarism *source* code LISP berbasis web yang menerapkan algoritma Rabin Karp. Langkah-langkahn deteksi kesamaan *source code* LISP adalah sebagai berikut:

a. Mengunggah berkas pertama.
b. Mengambil *string* dari berkas pertama.
c. Semua spasi dihilangkan sehingga semua karakter saling berhimpitan.
d. Dari tiap gram, dihitung nilai *hash* tiap gram menggunakan *rolling-hash.*
e. Dari *hash* yang dihitung, akan didapati *fingerprint* dokumen pertama.
f. Mengunggah berkas kedua.
g. Mengambil *string* dari berkas kedua.
h. Semua spasi dihilangkan sehingga semua karakter saling berhimpitan.
i. Dari tiap gram, dihitung nilai *hash* tiap gram menggunakan *rolling-hash.*
j. Dari *hash* yang dihitung, akan didapati *fingerprint* dokumen kedua.
k. Dihitung persentase tingkat kesamaan dokumen pertama dengan dokumen kedua menggunakan Dice Similarity Coefficient.
l. Persentase kemiripan didapat.

A. Pengertian Algoritma Rabin Karp

Algortima Rabin-Karp adalah algoritma pencarian *string* yang ditemukan oleh Michael Rabin dan Richard Karp. Karakteristik dari algoritma ini adalah 1) menggunakan fungsi hashing 2) fase *preprocessing* menggunakan kompleksitas waktu O(m) 3) Untuk fase pencarian kompleksitasnya adalah O(mn) 4) Waktu yang diperlukan : O(m+n). Prinsip algoritma Rabin-Karp adalah melakukan *hashing* untuk menghitung nilai pola dan menemukan potongan *string* dalam suatu teks, jika nilai *hash* yang didapat berbeda dari nilai *hash* yang diujikan, maka algoritma akan menghitung lagi *hash* dari teks sepanjang m. *Hash* adalah sekumpulan nilai dengan panjang tetap yang berasal dari teks dengan panjang teks beragam dan biasa digunakan otentifikasi data [11]. Fungsi untuk menghasilkan nilai hash disebut fungsi hash, sedangkan nilai yang dihasilkan disebut nilai *hash.* Dalam mencari kesamaan *string*, algoritma Rabin-Karp menggunakan *rolling hash* dalam menghasilkan *fingerprint* dari tiap *substring* dalam suatu *string* sebanyak k karakter. *String* inputan akan dikelompokkan menjadi *substring* dengan panjang m karakter, kemudian *substring* berikutnya didapat dengan cara maju 1 karakter dari *substring* sebelumnya dan membuang karakter pertama dari *substring* sebelumnya. Tiap *substring* disebut K-Gram. Untuk menghitung kemiripan *string* inputan dengan *string* contoh, dapat digunakan *Dice Similarity Coeficients*

B. Pengertian Rolling Hash

Algoritma *Rabin-Karp* menggunakan *rolling hash* untuk mencari string yang sama. *Rolling hash* melakukan perhitungan *hash* dengan cara mengambil *substring* dalam suatu *string* sebanyak k karakter. *String* inputan akan dikelompokkan menjadi *substring* dengan panjang m karakter kemudian dihitung *hash*-nya berdasar nilai ASCII dari tiap karakter, rumus menghitung *hash* adalah

$$c_1 * b^{k-1} + c_2 * b^{k-2} * \ldots + c_{k-1} * b + c_k$$

Dimana c adalah nilai ASCII karakter , b adalah basis, dan k adalah banyak karakter. Untuk menghitung hash berikutnya adalah dengan mengurangkan nilai hash dengan perhitungan di karakter pertama kemudian menjumlahkan dengan perhitungan karakter setelah karakter terakhir pada substring yang sedang dihitung. Simulasi perhitungannya adalah misal diberikan inputan *string* "dian" dengan m sebanyak 3 karakter. Nilai *hash substring* pertama ("dia") adalah : 100 +105 + 97 = 304 , untuk menghitung *hash* dari *substring* selanjutnya , maka dapat dilakukan dengan cara
H = (d + i + a) – d + n = 304 – 100 + 110 = 314.

C. Pengertian Dice Similarity Coeficient

Untuk menghitung kemiripan dapat digunakan *Dice Similarity Coeficients* [12] dengan cara menghitung jumlah K-Gram yang digunakan pada kedua dokumen yang diuji. Nilai kemiripan tersebut dapat dihitung dengan rumus :

$$S = \frac{2C}{A + B}$$

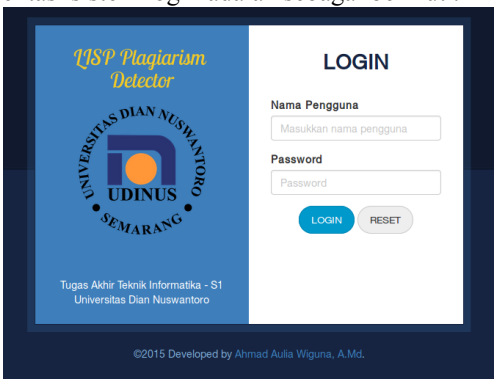
dimana S adalah nilai kemiripan, C adalah jumlah K-Gram yang sama dan A serta B masing-masing adalah jumlah K-Gram dari masing-masing *string*

yang diujikan. Hasil yang didapat adalah bilangan dari 0 sampai 100 dimana bilangan tersebut adalah persentase kemiripan 2 *string* yang diujikan.

III. IMPLEMENTASI

A. *Implementasi Login*

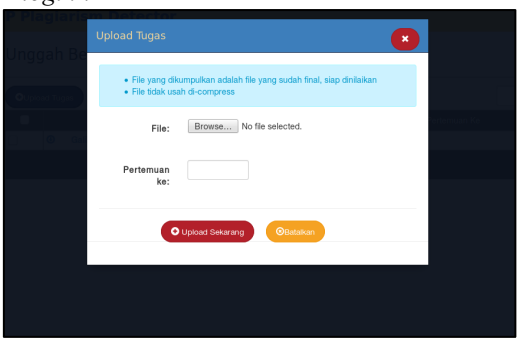
Halaman login merupakan halaman yang pertama kali dijumpai pengguna sebelum pengguna masuk ke sistem. Implementasi sistem login adalah sebagai berikut :



Gambar 1 Antarmuka Login

B. *Implementasi Unggah Source Code*

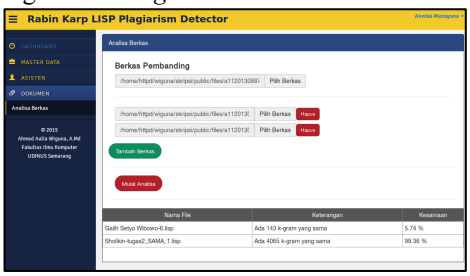
Berikut ini adalah implementasi halaman unggah *source code* yang dapat diakses oleh mahasiswa yang berhasil *login :*



Gambar 2 Antarmuka Unggah Berkas

C. *Implementasi Form Hitung Kemiripan*

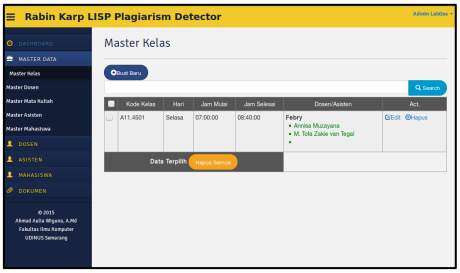
Berikut ini adalah implementasi *form* hitung persentase kemiripan yang dapat diakses oleh dosen atau asisten yang berhasil *login*:



Gambar 3 Form Hitung Persentase Kemiripan

D. *Implementasi Master Data*

Berikut ini halaman pengelolaan master data yang dapat diakses admin setelah berhasil *login*:



Gambar 4 Form Pengelolaan Master Data

IV. HASIL DAN PEMBAHASAN

A. *Hasil Perhitungan Hash Oleh Sistem Dibandingakan dengan Hasil Perhitungan Manual*

Dibawah ini merupakan perbandingan hasil perhitungan *hash* yang dilakukan oleh sistem dibandingkan hasil perhitungan yang dilakukan manual

Tabel 1 Tabel Perbandingan Hasil Perbandingan

No	*Substring*	Perhitungan *hash* oleh sistem	Perhitungan *hash* secara manual
1	(defu	511237	511237
2	defun	1112480	1112480
3	efuna	1124897	1124897
4	funad	1149070	1149070
5	unadd	1290800	1290800
6	nadd(	1208040	1208040
7	add(x	1080520	1080520
8	dd(xy	1105321	1105321
9	d(xy)	1053251	1053251
10	(xy)(	532550	532550
11	xy)(+	1325543	1325543
12	y)(+x	1255550	1255550
13	)(+xy	455621	455621
14	(+xy)	456251	456251
15	+xy))	562551	562551

Dari hasil di atas terlihat bahwa hasil perhitungan yang dilakukan sistem sudah sesuai dengan hasil perhitungan manual

B. *Hasil Ujicoba Penerapan Algoritma Rabin Karp pada Source Code LISP*

Hasil Uji Coba Penerapan Algoritma *Rabin-Karp* dalam menghitung persentase kesamaan 2 atau lebih berkas LISP dapat dilihat pada gambar berikut :

Gambar 5 Hasil Uji Coba Penerapan Algoritma Rabin-Karp

Dari gambar diatas dapat disimpulkan bahwa berkas yang diuji memiliki kemiripan sebesar 99,03 % terhadap berkas pembanding. Isi berkas pembanding dan berkas yang diuji adalah sebagai berikut :

```
(defun typePemb(n)
    (if(integerp n) t nil)
)

(defun typePeny(n)
    (if(and (integerp n) (not (equal n 0))) t nil)
)

(defun buatP(x y)
    (if (and (typePemb x) (typePeny y))(cons x (cons y nil)) nil)
)

(defun pemb(p)
    (car p)
)

(defun peny(p)
    (car (cdr p))
)

(defun tambahkanP(p1 p2)
    (/ (+ (* (peny p2) (pemb p1)) (* (peny p1) (pemb p2))) (* (peny p1) (peny p2)))
)

(defun SubP(p1 p2)
    (/ (- (* (peny p2) (pemb p1)) (* (peny p1) (pemb p2))) (* (peny p1) (peny p2)))
)

(defun mulP(p1 p2)
    (/ (* (pemb p1)(pemb p2)) (* (peny p1) (peny p2)))
```

Gambar 6 Isi Berkas Pembanding

```
(defun typePembilang(n)
    (if(integerp n) t nil)
)

(defun typePeny(n)
    (if(and (integerp n) (not (equal n 0))) t nil)
)

(defun makeP(x y)
    (if (and (typePembilang x) (typePeny y))(cons x (cons y nil)) nil)
)

(defun pemb(p)
    (car p)
)

(defun peny(p)
    (car (cdr p))
)

(defun addP(p1 p2)
    (/ (+ (* (peny p2) (pemb p1)) (* (peny p1) (pemb p2))) (* (peny p1) (peny p2)))
)

(defun SubP(p1 p2)
    (/ (- (* (peny p2) (pemb p1)) (* (peny p1) (pemb p2))) (* (peny p1) (peny p2)))
)

(defun mulP(p1 p2)
    (/ (* (pemb p1)(pemb p2)) (* (peny p1) (peny p2)))
```

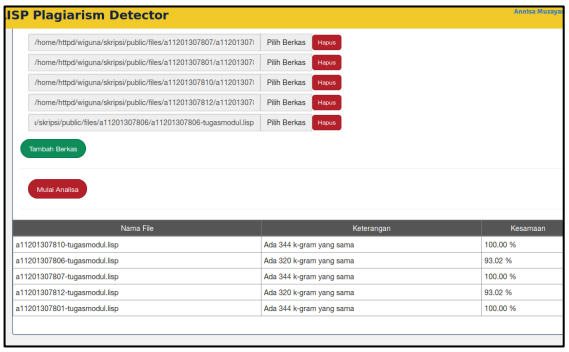
Gambar 7 Isi Berkas Yang Dibandingkan

Sedangkan dalam pengujian terhadap berkas LISP tugas mahasiswa, berkas yang digunakan adalah berkas dari mahasiswa kelas A11.4213 dan nama berkas yang diuji adalah adalah "tugasmodul.lisp" pada modul 2 , sedangkan mahasiswa yang berkasnya dijadikan acuan adalah pekerjaan mahasiswa dengan NIM A11.2013.07796. Kemudian berkas-berkas yang diuji adalah sebagai berikut :

Tabel 2 Tabel Pengujian Berkas

NIM	Nama berkas	Persentase kemiripan
A11.2013.07807	tugasmodul.lisp	100 %
A11.2013.07801	tugasmodul.lisp	100 %
A11.2013.07810	tugasmodul.lisp	100 %
A11.2013.07812	tugasmodul.lisp	93.02 %
A11.2013.07806	tugasmodul.lisp	93.02 %

Kemudian antarmuka yang ditampilkan aplikasi saat pengujian adalah sebagai berikut :



Gambar 8 Antarmuka Saat Pengujian Berkas

C. Hasil Pengujian Black Box

Pengujian black-box dilakukan untuk menguji apakah sistem telah berfungsi sesuai kebutuhan fungsional sistem. Berikut adalah hasil pengujian black-box pada sistem deteksi plagiarism berkas LISP.

Tabel 5. 3 Hasil Pengujian *Black Box*

	Kasus Uji	Output yang diharapkan	Output	Hasil
1	Login	Menampilkan halaman *login*	Senampilkan halaman *login*	Sesuai
2	Halaman utama mahasiswa	Menampilkan halaman beranda mahasiswa	Sistem menampilkan halaman beranda mahasiswa	Sesuai
3	Halaman utama dosen	Menampilkan halaman beranda dosen	Sistem menampilkan halaman beranda dosen	Sesuai
4	Halaman utama asisten	Menampilkan halaman beranda asisten	Sistem menampilkan halaman beranda asisten	Sesuai
5	Halaman utama admin	Menampilkan halaman admin	Sistem menampilkan halaman beranda admin	Sesuai
6	Halaman master dosen	Menampilkan halaman master dosen	Sistem menampilkan halaman master dosen	Sesuai
7	Halaman master asisten	Menampilkan halaman master asisten	Sistem menampilkan halaman master asisten	Sesuai

Lanjutan

No	Kasus Uji	Output yang diharapkan	Output	Hasil
13	Aktivasi asisten	Menampilkan halaman aktivasi asisten	Sistem menampilkan halaman aktivasi asisten	Sesuai
14	Unggah berkas	Menampilkan halaman unggah berkas	Sistem menampilkan halaman unggah berkas dan berkas yang diunggah berhasil disimpan di server.	Sesuai
15	Analisis berkas	Menampilkan halaman analisis berkas	Sistem menampilkan halaman analisis berkas dan menampilkan hasil perhitungan di tabel yang disediakan.	Sesuai

V. PENUTUP

A. Kesimpulan

Kesimpulan yang dapat didapat dari pembahasan atas hasil penelitian tugas akhir ini adalah sebagai berikut :

1. Sistem deteksi plagiarism kode sumber lisp dibangun sebagai aplikasi berbasis web untuk memudahkan pengguna dalam mengakses sistem, pengguna hanya membutuhkan koneksi intranet atau internet untuk dapat mengakses dan menggunakan sistem.
2. Sistem deteksi plagiarism kode sumber lisp yang dibangun dapat menghasilkan persentase kemiripan antara dokumen-dokumen yang diuji dengan dokumen yang dijadikan contoh. Persentase yang dihasilkan dapat digunakan oleh dosen maupun asisten dosen sebagai pertimbangan dalam memberi nilai pada mahasiswa yang bersangkutan.
3. Sistem deteksi plagiarism kode sumber lisp yang dibangun menerapkan algoritma Rabin-Karp dalam melakukan pencocokan *string* antara dokumen-dokumen yang diuji dengan dokumen yang dijadikan contoh. Tahapan pencocokan diawali dengan menghilangkan spasi pada *string*, membagi *string* ke *substring-substring* , menghitung *hash* tiap *substring* menggunakan teknik *rolling hash* kemudian menghitung persentase kemiripan antar dokumen menggunakan *Dice Similarity Coeficient* .

B. Saran

Sistem deteksi plagiarisme kode sumber lisp yang dibangun ini masih memiliki beberapa kekurangan dan kelemahan yang dapat dikembangkan dalam penelitian selanjutnya. Saran bagi penelitian selanjutnya yaitu sebagai berikut :

1. Sistem deteksi plagiarisme kode sumber lisp yang dibangun ini menghitung persentase kesamaan antar berkas hanya berdasar kandungan karakter-karakter di dalam berkas, belum memeriksa model *coding* dari masing-masing berkas, untuk itu dalam penelitian selanjutnya disarankan untuk menggabungkan deteksi plagiarism menggunakan algoritma Rabin_Karp dengan teknik deteksi berdasar model *coding* seperti dalam penelitian Sevtian Agus Prasetyo yang berjudul Deteksi Plagiarisme Source Code berbasis SIM dengan Metode Perbaikan *Coding Models Algorithm*.
2. Sistem deteksi plagiarisme kode sumber lisp ini menerapkan algoritma Rabin-Karp untuk melakukan pencocokan string namun persentase yang dihasilkan oleh sistem ini sangat bergantung pada banyaknya karakter selain spasi dalam sebuah berkas. Semakin banyak *string* dalam sebuah berkas maka hasil perhitungan akan semakin akurat.

REFERENCES

[1] Saniman dan Muhammad Fathoni (2008). *Pengantar Algoritma dan Pemrograman*. Jurnal SAINTIKOM Volume 4 Nomor 1. LPPM-STMIK Triguna Dharma

[2] Liem, Inggriani. 2007 .*Diktat Kuliah Dasar Pemrograman (Bagian Pemrograman Prosedural*. Sekolah TInggi Elektro dan Informatika Institut Teknologi Bandung

[3] Prasetyo, Sevtian Agus. 2013 .*Deteksi Plagiarisme Source Code berbasis SIM dengan Metode Perbaikan Coding Models Algorithm*. Skripsi FIK Udinus Semarang

[4] Firdaus, Hari Bagus. 2008 .*Deteksi Plagiat Dokumen Menggunakan Algoritma Rabin-Karp*. Jurnal Ilmu Komputer Dan Teknologi Informasi, Vol III No.2, Oktober 2003. Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung

[5] Diana Purwitasari, Putu Yuwono, Umi Laili Yuhana. *Deteksi Keberadaan Kalimat Sama sebagai Indikasi Penjiplakan dengan Algoritma Hashing Berbasis N-Gram*. Lab Semantik Web - Teknik Informatika, Institut Teknologi Sepuluh Nopember Surabaya

[6] Republik Indonesia. 2010. *Peraturan Menteri Pendidikan Nasional Republik Indonesia Nomor 17 Tahun 2010*. Kementerian Pendidikan Nasional

[7] Suwarjo, dkk. 2012. *Identifikasi Bentuk Plagiat Pada Skripsi Mahasiswa Fakultas Ilmu Pendidikan Universitas Negeri Yogyakarta*. Yogyakarta. Universitas Negeri Yogyakarta. Artikel Penelitian

[8] Harris, Simon. And James Ross (2006). *Beginning Algorithms*.. Willey Publishing, Indianapolis Indiana

[9] Vina Sagita dan Maria Irmina Prasetiyowati. 2013. *Studi Perbandingan Implementasi Algoritma Boyer-Moore, Turbo Boyer-Moore, dan Tuned Boyer-Moore dalam Pencarian String*. Ultimatics Vol 4 No. 1, Juni 2013. Universitas Multimedia Nusantara Tangerang

[10] Liem, Inggriani (2008). *Diktat Kuliah Dasar Pemrograman (Bagian Pemrograman Fungsional*. Sekolah TInggi Elektro dan Informatika Institut Teknologi Bandung

[11] V.R. Kulkarni, Shaheen Mujawar, and Sulabha Apte (2010). *Hash Function Implementation Using Artificial Neural Network*. International Journal on Soft Computing Vol. 1 No. 7, November 2010

[12] Sahriar Hamza, M.Sarosa dan Purnomo Budi Santoso (2013). *Sistem Koreksi Soal Essay Otomatis dengan Menggunakan Metode Rabin Karp*. Jurnal EECCIS Vol. 7 No. 2, Desember 2013

[13] Pressman, Roger S (2001). *Software Engineering A Practitioner's Approach*. The McGraw-Hill Companies. Inc New York

[14] Wahyudie, Firman (2008). *Pembuatan dan Perancangan Aplikasi E-Learning Sebagai Alat Bantu Perkuliahan.* Skripsi Teknik Informatika Fakultas Ilmu Komputer Udinus Semarang